Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.8
DOI: https://doi.org/10.24867/14BE08Gajic

RAZVOJ Kubernetes KLASTERA ZA RAD SA Docker KONTEJNERIMA

DEVELOPMENT OF CUBERNETES CLUSTERS FOR WORKING WITH DOCKER CONTAINERS

Nikola Gajić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *Zadatak rada predstavlja razvoj Kubernetes klastera za rad sa Docker kontejnerima na Raspberry Pi računarima. Za demonstraciju rada samog klastera, biće podignut Nginx veb server kao i load balancer radi pristupanja sadržaju Nginx veb servera. Projekat će biti realizovan uzupotrebom Docker-a, Kubernetes-a, Nginx kao i load balancer-a. Rešenje će biti predstavljeno uz pomoć curl komande iz Raspberry Pi komandne linije kao i iz dva različita veb pretraživača.*

Ključne reči: *Raspberry Pi, Docker, Kubernetes, Load Balancer, Nginx*

Abstract – *This paper presents one solution of developing Kubernetes clusters for working with the Docker container on Raspberry Pi platform. To demonstrate the work of the cluster, an Nginx web server and a load balancer will be set up to access the contents of the Nginx web server. The project will be realized using, Docker, Kubernetes, Nginx and Load balancer. The solution will be presented with the help of a curl command from the Raspberry Pi terminal as well as from two different web browsers.*

Keywords: *Raspberry Pi, Docker, Kubernetes, Load Balancer, Nginx*

1. UVOD

Zadatak rada predstavlja kreiranje *Kubernetes* klastera uz pomoć *Dockera* na tri fizička računara *Raspberry Pi 4 model B*. Kao primer za primenu klastera podignut je *Nginx* veb server i implementiran je *Load balancer* kojim se vrši demonstracija prisupa *Nginx* veb serveru.

Prvo poglavlje je uvodno, dok se u drugom nalazi opis izabranih tehnologija koje su korišćene pri izradi master rada. Treće poglavlje predstavlja detaljan opis konkretne implementacije dok je u četvrtom poglavlju iznet zaključak.

2. OPIS KORIŠĆENIH TEHNOLOGIJA

Za implementaciju čitavog sistema korišćeno je tri fizička *Raspberry Pi* računara, *Docker* za isporuku softvera i *Kubernetes* kao sistem za orkestraciju softvera otvorenog koda.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Milan Vidaković, red. prof.

Za samu demonstaciju rada klastera podignut je *Nginx* veb server i implementiran je *load balancer* service koji vrši proces raspodele zadataka sa ciljem da obrada samih zadataka bude što efikasnija.

2.1. *Raspberry Pi*

Raspberry Pi [1] predstavlja računar malih dimenzija koji je prvobitno bio namenjen za promociju podučavanja osnovnih računarskih nauka u školama.

2.2. *Docker*

Docker [2] predstavlja skup proizvoda koji koriste virtuelizaciju na nivou operativnog sistema za isporuku softvera u paketima koji se nazivaju kontejneri. Kontejner [3] je proces koji se odvija na samom računaru i koji može da komunicira sa ostalim procesima pokrenutim na istom računaru. *Docker* je moguće podesiti na različim arhitekturama što je prikazano u tabeli 1.

Tabela 1. *Podržane Docker arhitekture* [4]

Platform	*x86_64 / amd64*	*ARM*	*ARM64 / AARCH64*
CentOS	✓		✓
Debian	✓	✓	✓
Fedora	✓		✓
Raspbian		✓	✓
Ubuntu	✓	✓	✓

2.3 *Kubernetes*

Kubernetes [5] je sistem otvorenog koda za orkestraciju kontejnera kako bi se postigao određeni nivo automatizacije, skaliranja i upravljanja računarskim aplikacijama. *Kubernetes* klaster predstavlja skup računara koji su međusobno povezani i kao takvi se nazivaju nodovi [6]. Svaki klaster mora da ima najmanje jedan radni nod. Na nodovima koji se nalaze u klasteru se nalaze raspoređeni *pod*-ovi koji predstavljaju komponente same aplikacije.

Svaki *Kubernetes* klaster može da sadrži tri komponente:

- *kubelet*
- *kube-proxy*
- *Container runtime*

2.4 *Nginx*

Nginx [7] je veb server otvorenog koda koji takođe može da se ponaša kao *reverse-proxy, load balancer i mail proxy*. *Nginx*-ova modularna arhitektura zajedno sa asinhronim pristupom obrade zahteva može da pruži visoke performanse pod velikim opterećenjem.

2.5. *Load balancer*

Pojam *load balancer* [8,9] u računarstu se odnosi na proces raspodele skupa zadataka sa ciljem da ukupna obrada svih pristiglih zahteva bude što efikasnija. *Load balancer* izbegava problem neravnomernog raspoređivanja zahteva računarskim jedinicima kako ne bi došlo do preopterećenja sistema.

3. OPIS IMPLEMENTACIJE

Projekat je realizovan na tri *Raspberry Pi* računara na kojim je implementiran *Docker* sa *Kubernetes* klasterom, sa dodatnim servisom poput *load balancera* i *Nginx* veb serverom.

Na samom početku kreirane su tri *boot*-abilne SD kartice sa *Rasberry Pi OS*-om za sva tri fizička *Raspberry Pi*-a. Kreiranje boot-ablinih kartica izvedeno je uz pomoć *Raspberry Pi* Imager-a v1.6 nakon čega su iste pokrenute na samim *Raspberry Pi*-evima. Prilikom pokretanja *Raspberry Pi*-eva na svakom od njih je postavljena statička *IP* i to redom:

- 192.168.1.9,
- 192.168.1.10 i
- 192.168.1.11.

Pored postavljanja statičkih *IP* adresa izvršena je i promena imena (*hostname*-a) svakog od *Raspberry Pi*-a i to:

- *k8s-master*,
- k8s-worker01 i
- k8s-worker02.

Na *Raspberry Pi*-u postoji komandna linija koja je definisana u fajlu na *boot* particiji i naziva se *cmdline.txt*. Čitav fajl *cmdline.txt* služi za prosleđivanje argumenata *Linux Kernel*-u kako bi sam sistem startovao sa posebnim parametrima. Komandna linija je izmenjene i dodeljeni su joj parametri

- *cgroup_enable=cpuset*,
- *cgroup_enable=memory*,
- *cgroup_memory=1 i*
- *swapaccount=1*.

Na svakom *Linux* operativnom sistemu postoji deo memorije koji se koristi kada je količina fizičke *RAM* memorije puna, odnosno kada *Linux*-u ponestane *RAM* memorije neaktivni sadržaj se premešta iz *RAM* memorije u prostor za razmenu koji se naziva *SWAP*. Prostor za razmenu sadržaja je onesposobljen nakon čega je izvršeno ponovno pokretanje *Raspberry Pi*-a.

Nakon onesposobljavanja prostora za razmenu sadržaja započeta je instalacija *Docker*-a komadnom prikazanom na listingu 1.

```
curl -sSL get.docker.com | sh
```

Listing 1. Instalacija *Dockera*

Kako je *Docker* uspešno instaliran na *Raspberry Pi*-u potrebno je pridružiti *Docker* grupu na korisnika, kako bi sam korisnik mogao da vrši izmene na *Docker*-u. Kako je predefinisani korisnika na *Raspberry Pi*-u pi, upravo tom korisniku je potrebno dodati posebnu grupu, *Docker* grupu, i dozvoliti mu korišćenja *Docker*-a. Dodavanje nove grupe pi vrši se komandom koja je prikazana na listingu 2:

```
sudo usermod -aG docker pi
```

Listing 2. Dodavanje *Docker* grupe pi korisniku

Nakon dodavanja *Docker* grupe na listu grupa kojima pripada pi korisnik, izvršena je modifikacija daemon.json datoteke koji se nalazi na putanji */etc/docker/-daemon.json*. *Daemon.json* predstavlja servis koji se pokreće na samom operativnom sistemu i pokreće se zajedno sa sistemom, što rezultuje automatskim pokretanjem *Dockera* zajedno sa sistemom. Konfiguracija *daemon.json* datoteke prikazana je na listingu 3:

```
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
```

Listing 3. Konfiguracija *daemon.json* datoteke

Pored konfigurisane *daemon.json* datoteke, aktiviran je *routing*, odnosno usmeravanje internet saobraćaja. Prosleđivanje internet saobraćaja treba da bude omogućeno kada želimo da sam sistem funkcioniše kao ruter, odnosno da prenosi saobraćaj iz jedne mreže u drugu.

Nakon podešenog *Dockera* potrebno je dodati *Kubernetes repository* komandama prikazanim na listingu 4:

```
sudo nano /etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
```

Listing 4. Dodavanje *Kubernetes* skladišta

Takođe je potrebno dodati i *GPG key* (*GNU Privacy Guard*), koji predstavlja kriptografski paket koji omogućava korisnicima da razmenjuju šifrovane poruke. Dodavanje *GPG* ključa prikazano je na listingu 5:

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add –
```

Listing 5. Dodavanje pouzdanog ključa korisniku *pi*

Nakon što je dodat ključ kojim se omogućava komunikacija, izvršena je instalacija dodatnih *Kubernetes* alata za kreiranje *Kubernetes* klastera, uz pomoć komande prikazane na listingu 6:

```
sudo apt install kubeadm kubectl kubelet
```

Listing 6. Instalacija dodatnih alata

Sve prethodno definisane komande su se izvršavale uporedo na sva tri *Raspberry Pi*-a, čime je dobijeno da se na sva tri *Raspberry Pi*-a nalazi identična konfiguracija. Sledeći korak u konfiguracije je određivanje glavnog node u klasteru i izvršavanje komande predstavljene u listingu 7:

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```

Listing 7. Postavljanje glavnog čvora

Nakon izvršavanje komande na glavnom čvoru, u odgovoru dobijamo sledeće korake koje je potrebno izvršiti na glavnom čvoru kao i konekcioni string za preostale čvorove u klasteru. Koraci i konekcioni string su prikazani u listingu 8:

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
kubeadm join 192.168.1.9:6443 --token wqxip1.kccep08271rz0zxk --discovery-token-ca-cert-hash sha256:3ed2ebe3b6fff62d79311c5a7b1036f6949c6459a01ddfd7587cd60c8c503c35
```

Listing 8. Odgovor nakon init komande na glavnom računaru

Umrežavanje predstavlja centralni deo svakog *Kubernetes* klastera. Postoji nekoliko načina za implementaciju umrežavanja, dok je za potrebe ovog projekta korišćen flannel. *Flannel* [10] je jednostavna mreža koja zadovoljava sve zahteve *Kubernetesa*. *Flannel* se instaliran komandom koja je prikazana na listingu 9:

```
kubectl apply –f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

Listing 9. Instalacija *Flannela*

Nakon instalacije *Flannela*, potrebno je dodati i preostale čvorove u *Kubernetes* klaster komandom koja je predstavljena u listingu 8.

Proveru rada samog *Kubernetes* klastera i svih njegovih aspekata je moguće izvršiti iz komande linije glavnog računara u klasteru. Da bi proverili da li su čvorovi validno dodati u klaster kao i da li su operativni, koristi se komanda prikazana u listingu 10:

```
kubectl get nodes –o wide
```

Listing 10. Provera čvorova u klasteru

Na samim čvorovima, nakon njihovog dodavanja u klaster se ne nalazi ništa, prazni su. Potrebno je kreirati *deployment*, odnosno aplikaciju koje će se nalaziti na *pod*-ovima koji se nalaze na čvorovima na računarima. Kreiranje *deployment*-a je prikazano na listingu 11:

```
cat <<EOF | kubectl create -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
```

Listing 11. Kreiranje *nxinx deployment*-a

Provera ispravnosti instalirane aplikacije realizuje se komandom predstavljenom u listingom 12:

```
kubectl get pods –o wide
```

Listing 12. Provera *deploymenta* na *pod*-ovima

Nakon instalacije, da bi ceo sistem funkcionisao na ispravan način, potrebno je kreirati *load balancer*. Za kreiranje *load balancer*-a korišćen je *MetalLB* koji predstavlja *load balancer* za *Kubernetes* klastere koji koristi standardne protokole za mrežno usmeravanje. Instalacija *MetalLB*-a se izvodi pokretanjem komandi iz komandne linije koje su prikazane na listingu 13:

```
kubectl apply –f
https://raw.githubusercontent.com/metallb/metallb/v0.9.6/manifests/namespace.yaml
kubectl apply -f
https://raw.githubusercontent.com/metallb/metallb/v0.9.6/manifests/metallb.yaml
kubectl create secret generic -n metallb-system memberlist --from-literal=secretkey="$(openssl rand -base64 128)"
```

Listing 13. Instalacije *MetalLB*-a

Proveru da li je namespace za *MetalLB* kreiran vrši se komandom prikazana na listingu 14:

```
kubectl get namespace -o wide
kubectl get all –n metallb-system
```

Listing 14. Provera svih *namespace*-a

Nakon same instalacije *MetalLB* potrebno je podesiti *ConfigMap*-u na čitavom sistemu. *ConfigMap*-e omogućavaju da se razdvoji specifična konfiguracija kontejnera što čini aplikaciju prenosivom. Kreiranje *ConfigMap*-e predstavljena je na listingu 15:

```
cat <<EOF | kubectl create -f -
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    address-pools:
    - name: default
      protocol: layer2
      addresses:
```

```
    - 192.168.1.240-192.168.1.250
EOF
```

Listing 15. Kreiranje *ConfigMap*-e

Poslednji korak u kreiranju *load balancer*-a realizuje se komandom koja je predstavljana na listingu 16:

```
kubectl expose deploy nginx-deployment --port 80 --type Load balancer
```

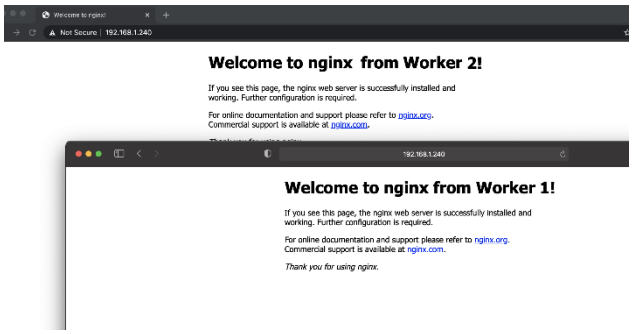
Listing 16. Kreiranje *load balancer* servisa

Pregled kompletnog *load balancer* servisa moguće je izvšiti komandom koja je predstavljena na listingu 17:

```
kubectl get svc –o wide
```

Listing 17. Pregled *load balancer* servisa

Proveru ispravnosti sistema moguće je startovanjem dva veb pretraživača i kucanjem adrese 192.168.1.240, što je i prikazano na slici 1. Sa slike se vidi da je *load balancer* rasporedio zahteve sa dva pretraživača na dva fizička čvora.



Slika 1. *Provera ispravnosti sistema putem veb pretraživača*

4. ZAKLJUČAK

Osnovni zadatak ovog rada je bio razvoj i demonstracija *Kubernetes* klastera za rad sa *Docker* kontejnerima na fizičkim *Raspberry Pi* računarima. Osnovna ideja je bila kreiranje klastera od tri *Raspberry Pi* računara na kojima se nalazi *Nginx* veb server kojem se pristupa putem veb pretraživača.

Za demonstraciju rada klastera kreiran je *load balancer* kojim se pristupa putem veb pretraživača. U zavisnosti od računara u klasteru kojem je zahtev prosleđen vraća se sadržaj na veb pretraživač.

Projekat je razvijen u aktuelnoj tehnologiji i ima dobru osnovu za dalji razvoj. Proširenje je moguće u smislu dodavanja još fizičkih računara, odnosno *Raspberry Pi*-a kao i *deploy*-ovanje dodatnih aplikacija na klaster.

5. LITERATURA

[1] https://en.wikipedia.org/wiki/Raspberry_Pi (pristupljeno u maju 2021.)

[2] https://en.wikipedia.org/wiki/Docker_(software) (pristupljeno u maju 2021.)

[3] https://docs.docker.com/get-started/ (pristupljeno u maju 2021.)

[4] https://docs.docker.com/engine/install/ (pristupljeno u maju 2021.)

[5] https://en.wikipedia.org/wiki/Kubernetes (pristupljeno u maju 2021.)

[6] https://kubernetes.io/docs/concepts/overview/components (pristupljeno u maju 2021.)

[7] https://en.wikipedia.org/wiki/Nginx (pristupljeno u maju 2021.)

[8] https://en.wikipedia.org/wiki/Load_balancing_(computing) (pristupljeno u maju 2021.)

[9] https://www.nginx.com/resources/glossary/load-balancing/ (pristupljeno u maju 2021.)

[10] https://github.com/flannel-io/flannel#flannel (pristupljeno u maju 2021.)

Kratka biografija:

Nikola Gajić rođen je 19.12.1994. u Novom Sadu. Završio Osnovnu školu „Petar Kočić“ u Inđiji i „Gimnazija – Prirodno matematički smer“ u Inđiji. Završio je osnovne akademske studije na Fakultetu Tehničkih Nauka u Novom Sadu, smer Računarstvo i Automatika. Nakon završenih osnovnih akademskih studija upisao je master akademske studije na istom fakultetu, smer Računarstvo i automatika modul Primenjene računarske nauke i informatika.
kontakt: gajicnikola41@gmail.com